Pergamon

Computers chem. Engng Vol. 22, Suppl., pp. S847–S850, 1998

Printed in Great Britain
PII: S0098-1354(98)00163-X
0098-1354/98 $19.00 + 0.00

Combinatorial technique for short term scheduling of multipurpose batch plants based on schedule-graph representation

E. Sanmartí[1], F. Friedler[2], L. Puigjaner[1]

[1]Chemical Engineering Department. Universitat Politècnica de Catalunya.
E.T.S.E.I.B., Diagonal 647, E-08028 Barcelona, Spain.
[2]Department of Computer Science. University of Veszprém.
Veszprém, Egyetem u. 10, H-8200, Hungary.

Abstract
This paper deals with the production scheduling of multipurpose batch plants. A novel graph representation is proposed that takes into consideration the specific characteristics of chemical processes in scheduling. In this graphs, the nodes represent the production tasks and the arcs the precedence relationships among them. The representation is flexible enough to consider a great variety of production structures (recipes with branches, alternative units, ...). Both NIS and UIS transfer policies can be considered simply by choosing the appropriate precedence relationships. This representation provides the opportunity of incorporating highly efficient graph algorithms together with an appropriate branch-and-bound (B&B) algorithm for solving multipurpose scheduling problems effectively. The B&B algorithm takes care of the combinatorial optimization problem involved in each scheduling problem, while the graph algorithms allow to obtain the lower bounds that control the branching strategy. The efficiency of the proposed method is established by comparing it with the application of a generic B&B solving an equivalent MILP scheduling model.

INTRODUCTION

Multipurpose batch plants are characterized by their flexibility and capability of producing a large number of different products in different qualities (by client demand), and by the possibility of using alternate production paths for producing the same product. This high degree of flexibility establishes the complexity of the scheduling problem. Moreover, even the simplifying assumption of permutation schedules (Reklaitis, 1981), in which complete batches are sequenced instead of tasks, must be discarded, since feasible schedules that would eventually lead to good schedules could be ignored. A large number of variables have to be taken into account to attain an efficient operation of the plant: unit assignments, products and/or tasks sequencing and tasks timing.

Short term scheduling of chemical multipurpose batch plants has similarities with the job-shop scheduling problem, which has been widely treated in operations research. An approach that has been traditionally used is based on a graph representation of the scheduling problem combined with a B&B algorithm (Adams et al., 1988; Carlier and Pinson, 1989). The B&B algorithm is usually coupled with specific heuristics for the job-shop problem that greatly accelerate the convergence of the B&B to the optimal or near optimal solutions.

The scheduling problem in chemical batch plants substantially differs from the job-shop. The former is more complex than the latter for several reasons. For instance, the problem of intermediate storage of liquid materials in chemical processes may not appear in the discrete mechanical manufacturing industry. These differences prevent the direct applicability of the graph representation and algorithms developed for job-shop scheduling in batch chemical systems. Batch chemical systems are usually scheduled by using other techniques, mainly mathematical programming (Sanmartí et al., 1996), sequencing and scheduling via tailored heuristics or stochastic (simulated annealing, genetic algorithms) methods (Graells et al., 1996). In this work, the scheduling problem in a chemical multipurpose batch plant is solved using an appropriate graph representation and a branch-and-bound algorithm.

GRAPH REPRESENTATION OF PRODUCT MASTER RECIPES

The master recipes are represented as a directed conjunctive graph, where the nodes represent the production tasks and the arcs are the precedence relationships among tasks. An additional node is associated with each product: the last task or tasks of the production are connected to the corresponding node by an arc. Thus, for each product, the number of nodes in the graph is the number of tasks of the recipe plus one. The number above the arrows represents the processing time of the task. Figure 1 illustrates the conventional and Figure 2 the graph

representation of the recipes of an example with three products.

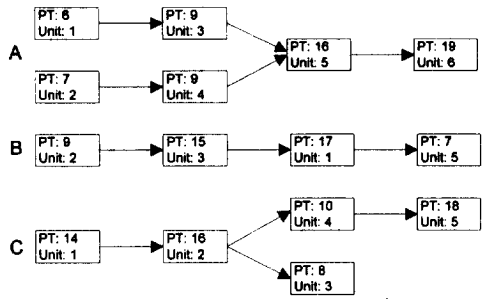

Figure 1. Conventional representation of the master recipes of three products (*PT*: processing time)

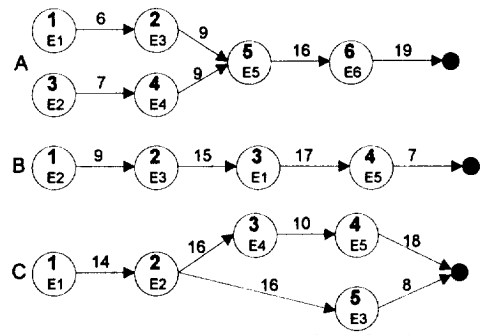

Figure 2. Graph representation of recipes

Each node of the graph contains the node number and the equipment unit assigned to the corresponding task (beginning with *E*).

GRAPH REPRESENTATION OF PRODUCTION SCHEDULES: *SCHEDULE-GRAPH*

Assume that the number of batches of each product to be produced is already known and a definite assignment of units to tasks is given. Then, the sequence of the tasks to be processed can be represented in a graph (Adams et al., 1988) where the task sequence of each unit is defined by a set of conjunctive arcs that connect the tasks assigned to the same unit. These arcs have a length equal to the processing time of the tasks, and they introduce a set of precedence constraints in addition to the constraints imposed by the recipe. For example, the task sequence of unit E1, 1-7-9, is given in Figure 3. It can be observed that task 7 is subject to two precedence constraints: it cannot start until task 6 is completed (recipe) and until task 1 has finished (sequence).

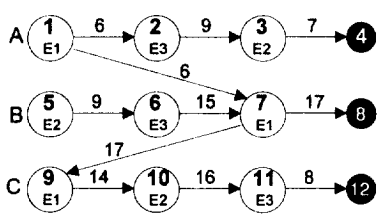

Figure 3. Conventional representation of a sequence for unit E1: convenient only for *UIS*

NIS transfer policy

The conventional graph representation given above is convenient in solving the job-shop problems with Unlimited Intermediate Storage (*UIS*) policy, however, it may not be useful for Non Intermediate Storage (*NIS*) cases. For the former, it is assumed that a unit is available each time that a task has been processed, i.e. the intermediate product that has been generated in this task is removed from the unit and stored somewhere until the following task in the recipe starts processing it. In chemical batch processes, however, the *NIS* transfer policy is usually followed. In this case, a unit is not free after processing a task until the material stored in it has been transferred to the unit assigned to the next task in the recipe. These additional constraints imposed by the *NIS* policy are expressed by additional arc or arcs. Suppose that unit *Ei* is assigned to task T_{jk}, then, to T_{lm}. Let N_{jk} denote the set of tasks that follow task T_{jk} according to the recipe. Then, a zero length arc (or an arc with the length of the changeover time if applicable) is established from each element of N_{jk} to T_{lm}. This representation is called schedule-graph. Note that while conventional representation is not convenient for *NIS*, schedule graph is able to represent both *NIS* and *UIS* scheduling problems correctly. The task sequence of unit *E1* given in Figure 2 is shown via schedule-graph on Figure 4. Instead of connecting tasks 1 and 7 and tasks 7 and 9 with arcs of length the processing time, zero length arcs are used to connect tasks 2 to 7 and 8 to 9, respectively.

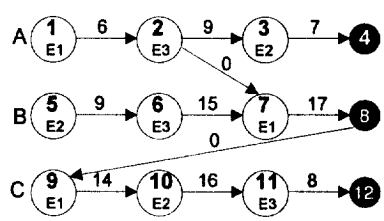

Figure 4. Schedule-graph representation of a task sequence for unit E1 with *NIS*

Feasible and infeasible schedules

Since a schedule represents a precedence in time and the corresponding schedule-graph represents precedence constraints, loops may not appear in a schedule-graph of a feasible schedule. A feasible schedule for the *UIS* transfer policy may be infeasible for the *NIS* case. This infeasibility can be detected in the schedule-graph representation, however, it may not be recognized on the conventional representation. The schedule given in Figure 5 for illustration is infeasible for *NIS* policy, its schedule-graph includes a loop expressing this fact.

Complete schedules

When all tasks have been sequenced for all the units, a complete schedule has been generated. Figure 6 contains the sequences of task that minimizes the

makespan for the example of the graph with *NIS* policy.

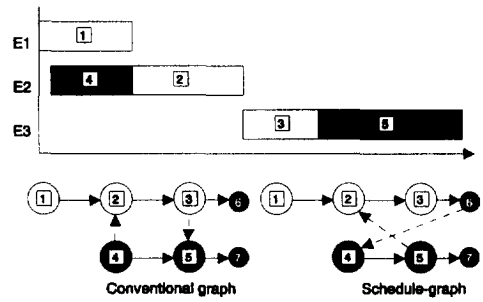

Figure 5. Representations of an infeasible schedule: schedule-graph expresses the infeasibility by a loop

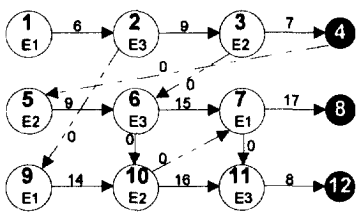

Figure 6. *NIS* sequence of a minimal makespan

Makespan calculation

The makespan of a schedule can be calculated directly on the schedule-graph using a so-called longest path algorithm. Since the complexity of this algorithm is polynomial, it can be applied effectively in solving scheduling problems. The longest path algorithm traces the graph backwards starting from the additional nodes (shown in black in the figures). This algorithm will determine the value of the longest path from each node to the sink node. The path with the maximal length expresses the makespan of the schedule. The start time of each task in the schedule can be determined by the longest path of this task. Thus, longest path algorithm also provides the timing of the whole schedule. Nevertheless, this timing is valid only if unlimited waiting times are allowed, i.e., if once the operation of a task is finished, the generated intermediate product can be hold in the unit for an unlimited time without stability problems of the materials. If this assumption is not valid, the longest path algorithm determines a *lower bound* for the makespan, instead of the real makespan. However, this lower bound is very useful for the application of a B&B algorithm, as it is shown later.

If limited waiting time is applied, the feasibility of a schedule or a partial schedule is checked using a linear programming (LP) model. The objective of this model is to minimize the makespan, while the constraints take into account the recipes and the concurrencies of the units among different tasks. The solution of this model gives the exact timing of all the operations provided that a feasible solution exists.

DETERMINATION OF THE OPTIMAL SCHEDULE

The optimal schedule is determined applying a branch-and-bound framework. Each node in the B&B tree corresponds to a partial schedule. At the root of the tree, the precedence constraints of the product recipes are applied only, i.e., the schedule-graph contains no arcs representing task sequences in the units. For this graph, the longest path algorithm can be applied to obtain a lower bound of the makespan. Then, the tasks assigned to the different units are sequenced one by one. Each time a task is sequenced, a branch is generated in the tree, and the longest path algorithm is applied again. When the tree reaches the bottom, a complete schedule has been obtained and an upper bound of the makespan can be calculated. As it has been mentioned previously, the longest path algorithm assumes that unlimited waiting times are allowed. If this is not the case, the upper bound of the makespan is calculated using a linear programming (LP) model.

Each time that the lower bound of the partial schedule of a node is greater than the current upper bound, or that the partial schedule is not acyclic (i.e. is infeasible), the branch that starts in the node is pruned. The scheme of this algorithm is given in Figure 7.

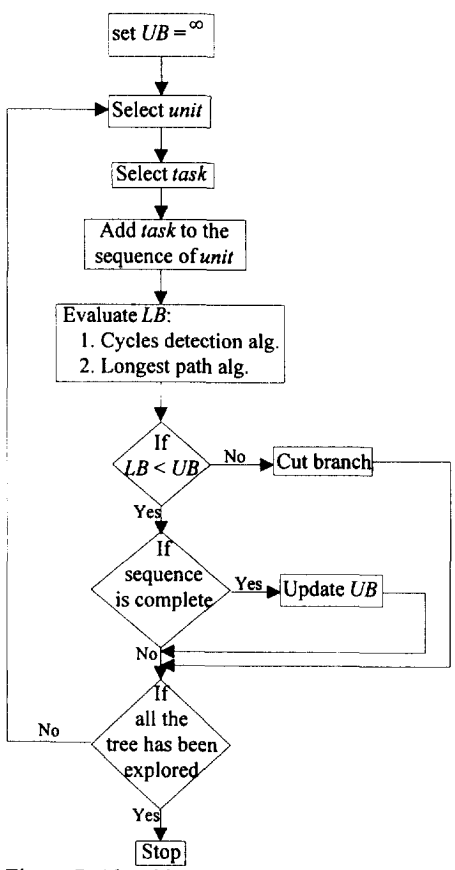

Figure 7. Algorithm for generating the optimal schedule

Considering more than one unit to perform a task
There is the possibility that more than one unit is available to perform a task. In this case, in addition to the task sequencing, the assignment of units to tasks has to be carried out previously. The B&B algorithm given above can be also adapted to control the units assignment.

More than one batch per product
Multipurpose scheduling problems usually treat different batches of the same product as different products. The adoption of this assumption in this case would lead to the generation of schedules that are essentially identical. The algorithm determines the set of batches that belong to the same product, and avoids the generation of those branches of the tree that would lead to the same solution. Two virtually identical schedules are given in Figure 8. The proposed algorithm generates only one of these schedules.

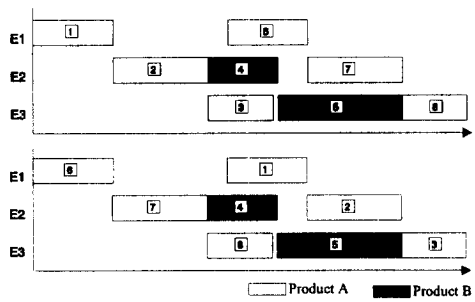

Figure 8. Repeated solutions

CASE STUDY
The algorithm has been applied to the solution of the following case study:
- 4 units: E1, E2, E3 and E4
- 4 products: A, B, C and D
The recipes of the products are given in Table 1, the waiting is limited to 20h for each intermediate product.

Table 1: Product recipes.

	Product A		Product B		Product C		Product D	
task	Unit	time	Unit	time	Unit	time	Unit	time
1	E1	6	E2	9	E4	8	E2	7
2	E3	9	E3	15	E1	14	E3	11
3	E4	7	E4	17	E2	16	E1	4

This problem has been solved for different number of batches using the proposed strategy and also using the MILP model presented in Sanmartí et al. (1996). The base problem consists the production of one batch for each product (4 batches). Then, additional batches are produced by repeating batches of product A, B, C and D (up to 8 batches). The MILP model has been solved with a generic solver (GAMS/OSL). The results are shown in Table 2 (PC-Pentium 200 Mhz). For the cases with 6, 7 and 8 batches, the MILP did not arrive to the optimal solution in reasonable time (in several hours), or the solver crashed. It can be seen that the proposed strategy obtains the optimal solution of the problem in a considerable lower CPU time.

Table 2: CPU time usage (s).

Batches	Proposed B&B	MILP	Makespan
4	0.12	8.24	47
5	1.93	380.00	62
6	12.76	-	73
7	71.09	-	87
8	1192.34	-	92

CONCLUSIONS
The proposed schedule-graph representation allows to represent scheduling problem using similar graph representations as those used to solve the job-shop problem but contemplating the higher complexity of the chemical multipurpose batch scheduling problems. If the problem is solved using a B&B, it is possible to use efficient graph algorithms to evaluate the makespan (or its lower bound) in each node of the enumeration tree instead of solving a relaxed MILP problem. Examples illustrate the efficacy of the proposed approach.

In contrast to the B&B techniques used to solve the job-shop problem, here no heuristics are used to guide the branching procedure. Thus, it seems that the embedding of appropriate heuristics to the chemical multipurpose batch scheduling problem would further enhance the method, which is the next step in our development.

ACKNOWLEDGEMENTS
The authors gratefully acknowledge financial support from the European Union under project JOE3-CT95-0036, as well as from the Comissió Interdepartamental de Recerca i Tecnologia under project QFN95-4702.

REFERENCES
Adams, J., E. Balas, D. Zawack, 1988, The shifting bottleneck procedure for job shop scheduling, *Management Science*, **34**, 391-401.
Carlier, J., E. Pinson, 1989, An algorithm for solving the job-shop problem, *Management Science*, **35**, 164-176.
Graells, M., A. Espuña, L. Puigjaner, 1996, Sequencing intermediate products: A practical solution for multipurpose production scheduling, *Computers chem. Engng.*, **20S**, S1137-S1142.
Reklaitis, G.V, 1981, Review of Scheduling of Process Operations, *AIChE Annual Meeting*, New Orleans, Nov..
Sanmartí, E., A. Espuña, L. Puigjaner, 1996, An MILP formulation for the production scheduling of multipurpose batch chemical plants, *7th Mediterranean Congress of Chemical Engineering*, Barcelona (Spain).